ITCS 6156

Machine Learning

Report on

Dog Breed Classifier

Team Members

Karthik Rao

Kiran Korey

Narahari Battala

Under the guidance of

Minwoo Jake Lee

UNC CHARLOTTE

DEPARTMENT OF COMPUTER SCIENCE

College of Computing and Informatics

UNC Charlotte, North Carolina – 28262

Spring 2018

Table of Contents

Table of Figures

1 Introduction

1.1 Problem Description

The objective of this project is to design and develop a Model/Algorithm that will classify the images of a dog to its specific breed by looking at its image. This is a fine-grained classification problem. The user will specify an image as input to the model and it will detect if it is a dog or not a dog. If a dog is detected in the image it will further detect the breed of the dog. This will serve as a pipeline to a mobile app or a web app.

1.2 Motivation

The great variety in dog's breed poses a significant problem to those who interested in identifying the breed of the dog, unless the dog falls into one of the very few widely known breeds it is difficult to identify the dog's breed. Though the problem is a classification problem, it is challenging compared to other classification problems as it’s a fine-grained classification because of the nature of the dog breeds. The difference between different dog breeds is very less and to make it more complex the difference between the dogs of same breeds is relatively very high. This fine-grained nature of dog breeds helps in building a better model which can also be further use to classify other animals to their respective breeds. This solution can also be used to in many other real-world scenarios like bio-diversity studies, identifying stray dogs, developing breed specific medicines. In this project we have used the images of dogs as there is a huge amount of dataset that is available and there is a huge variety of dog breeds (133 different types of dogs according to the dataset used here), if high accuracy is achieved these models give raise to wide variety of applications.

1.3 Review

There is plenty of work in the field of fine-grained image classification, one of the earlier works in fine-grained classification was an attempt at identifying plant species by Belhumer. This approach involved segmenting a leaf and then determining its species. A similar approach is followed by Farell to identify a bird’s species by finding key points along the beak, eyes, wings, feet and tail and building features around them.

More relevantly in 2012, a paper [1] from the University of Toronto was published. This paper mainly proposed to use a deep Convolutional Neural Network (CNN)for the task of image classification and developed architecture called AlexNet Architecture using deep CNN, it went on to become the most influential papers in the field. Paper [2] uses PCA, LDA & SVM to extract features and classify MRI images. The objective here is to detect brain tumor. The experimentation involved classification by SVM & the accuracy is said to be 98.87%. Another paper [3] says that CNN requires significant number of images as training data and substantial time for training and achieving higher accuracy for the classification. This demerit of CNN can be overcome by Transfer Learning. The proposed method uses the Inception model and classifies the dog images using the Tensorflow Library. The experimentation achieves an overall accuracy of 96%. Paper [4], this is a Stanford project which uses a combination of machine learning & image processing techniques to classify dog breeds. The project experiments using the following models: Bag of word, SVM, K-nearest neighbor, logistic regression. The paper concludes by saying that the SVM model is the best classifier with accuracy of 52%.

1.4 Open Questions

One of the major problems in image classification domain is the time and resources needed for training the model and the size of the dataset needed to train the model. This huge consumption of time and resource is because of large addition/multiplication operations on the image matrix which can be 2D,3D or 4D. Though **transfer learning** and **ImageNet** dataset solves the problem to a significant level there is still a lot of scope for improvement. The time to train can be reduced to certain extent by using GPU processors but again since GPU's are costly and there is a resource(money) associated to it.

1.5 Summary of Proposed Approach

1.5.1 Approach 1: (Most common approach)

In this approach, an already pre-trained deep neural network model which is trained using larger dataset of images from ImageNet is wired or attached with a new layer of classification i.e. several additional fully connected layers with the softmax layer on top of them. This approach is popularly known as transfer learning. The figure below will provide more insight to it.

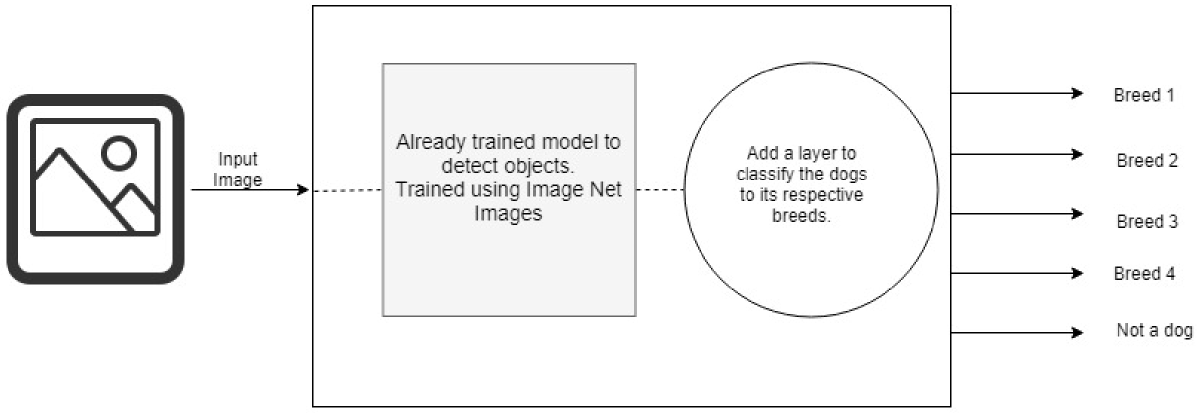

Input Image
Already trained model to detect objects. Trained using Image Net Images
Add a layer to classify the dogs to its respective breeds.
Breed 1
Breed 2
Breed 3
Breed 4
Not a dog

Figure 1: Transfer Learning Approach

In this approach the training must be done only on the original data set and training just the classification layer is much faster and easier compared to training from scratch.

In paper [3], Google's pre-trained Inception model on ImageNet dataset comprising of images of different classes and Transfer learning approach is followed to classify images.

1.5.2 Approach 2: (From Scratch)

This is one way of building a classifier that we thought about. In this approach, another bigger dataset of random images is added to the original dataset of images and the model i.e. CNN is trained from the scratch. The figure below will provide more insight to it.

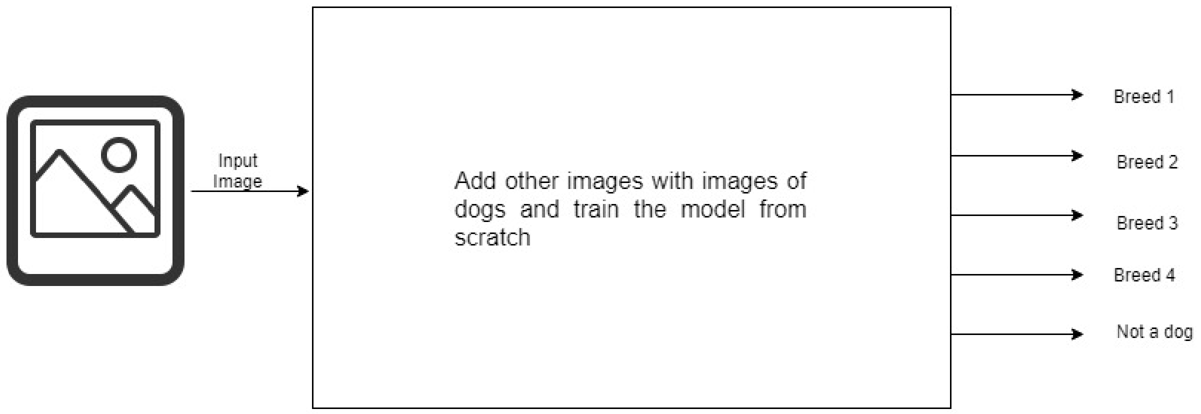

Input Image
Add other images with images of dogs and train the model from scratch
Breed 1
Breed 2
Breed 3
Breed 4
Not a dog

Figure 2:Basic Model

In this approach a much bigger amount of data must be analyzed and the training on this big dataset will take much more time and resources, so it is computationally heavy. Since time taken to train the model depends on the number of random images added to the dataset we cannot add less images to reduce the time it will reduce classifier accuracy significantly.

1.5.3 Approach 3: (Similar to Classifier Chain) (Our Approach)

Classifier chain: In this approach a multi-label classification is transformed into one or more single label classification and are chained together to form a classifier chain.

We slightly modified this approach and chained a binary classifier to a multiclass classifier. This is 2nd way of building a classifier that we thought about. In this approach a Binary classifier is chained with a Multiclass classifier. The binary classifier is a CNN model trained with random images and dogs image dataset but here the dogs image dataset will not be labeled according to its breeds, so it is just Dog – Not a Dog classifier. If the image is not of a dog the execution is halted and if the input image is of a dog it is passed on the multiclass classifier which is also a CNN model and the breed of the dog is predicted. The figure below will provide more insight to it.

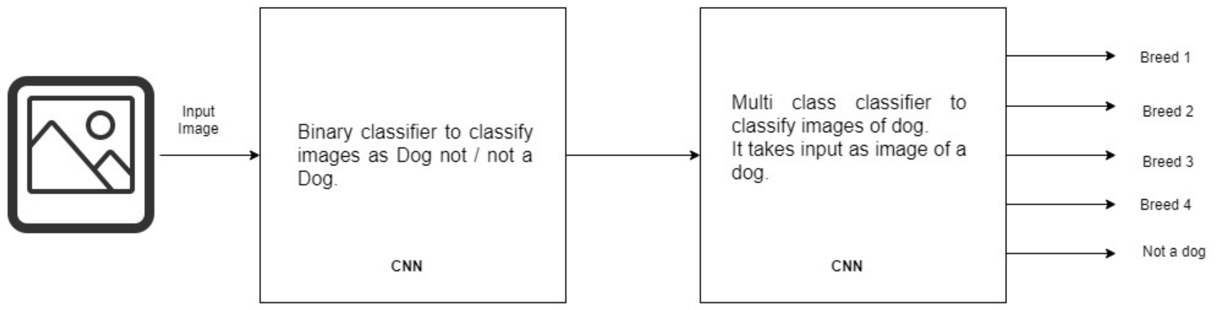

Input Image
Binary classifier to classify images as Dog not / not a Dog.
CNN
Multi class classifier to classify images of dog. It takes input as image of a dog.
CNN
Breed 1
Breed 2
Breed 3
Breed 4
Not a dog

Figure 3: Classifier Chain

In this model we can train both the models i.e. binary classifier and multiclass classifier parallelly as they are not dependent on each other. Paper [5] talks about talks about classifier chain technique.

2 Background

2.1 Survey of other Related Researches:

There are number of researches that are similar and related to Dog breed Identification. The most related approach to our work is described by Whitney LaRow, Brian Mittil and Vijay Singh in their paper, Dog Breed Identification. They used convolutional neural network to assist with key point detection in dogs, namely identifying eyes, nose and ears. As CNN's have seen success in identifying facial key points in humans, they have applied this technique to dogs as well. They have used Multinomial Logistic Regression, SVM and nearest neighbor models for classification. In this approach they trained a convolutional neural network on images of dogs and annotated facial key points and then they used the network to predict key points on an unseen test set of images. These predicted key points were then fed into a feature extraction system that used these key points to create more meaningful features from the image, which could later be used to classify the image.

They ran each of their classifiers using the SIFT descriptor feature set and compared the accuracy of each model, out of all the models the SVM and logistic regression models clearly produce the best results, obtaining 49% and 45% accuracy respectively. The nearest neighbor model performed at 22% accuracy.

In another paper, Transfer Learning for Image Classification of various dog breeds by Pratik Devikar , this approach uses a fine tuned model through transfer learning is used to perform classification on image data set. The main reason they have used transfer learning is due to draw backs of using convolutional neural networks, convolutional neural networks requires a significant number of images as training data and substantial time for training and achieving higher accuracy on classification. In this they have used Google's Inception-v3 model trained on the images and then used this model to achieve overall accuracy of 96%.

3 Method

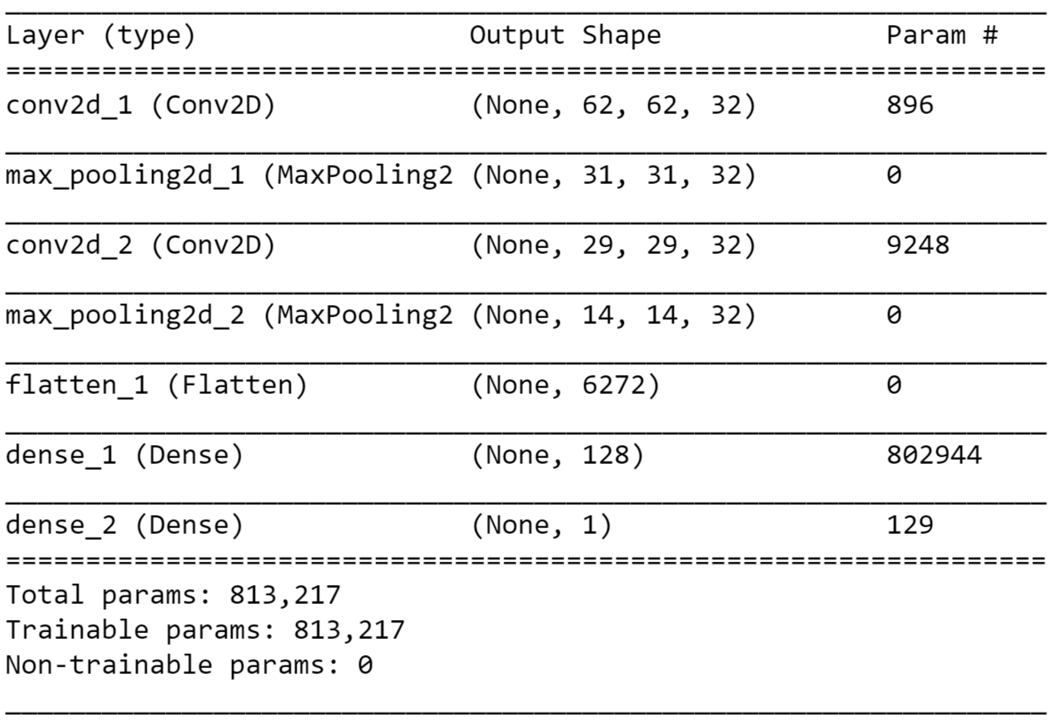
As mentioned earlier, to solve this fine-grained classification problem we have implemented two CNN models, one acts as a binary classifier which detects if image is dog or not dog and the other acts as a multiclass classifier which classifies the image of a dog to it's breed.

Let us see more in detail regarding these two CNN models.

3.1 Binary Classifier

The first step in classifying a dog according to it's breed is to detect if the image is a dog's image. So, for this we have built a Dog-Not-Dog model. We have created our own dataset for this by combining two different datasets, the dog dataset is from the dogs-cats dataset from Kaggle and not-dog data set is from ImageNet. ImageNet is a dataset of over 15 million labeled high-resolution images belonging to roughly 22,000 categories. The images were collected from the web and labeled by human labelers. The algorithms performance is highly dependent on the dataset, so we have applied data augmentation by randomly shifting or zooming or horizontally flipping images in our dataset using keras **ImageDataGenerator**. This increases the size of the training set, which will increase accuracy by decreasing overfitting.

Moving on to the CNN model for binary classifier we have used Sequential class from Keras models to build the CNN, we have two layers of convolution each paired with a Pooling layer and then the extracted features are flattened (horizontally stacked) and fed into the dense fully connected layer. Here we have 1 fully connected layer with 128 units and the last layer acts as output layer with just 1 unit. The convolution layer runs a sliding window through the image and at each step convolves the subimage with several filters i.e. each convolution layer extracts 32 features by using a stride (sliding window) of (3, 3), **relu** activation function is used to remove the negative values from these extracted features. To the first convolution layer the input image is formatted to the size (64,64) and since all the images are colored i.e. consists of RGB values we the complete size of the matrix becomes (64,64,3). The pooling layer reduces the size of the representation and the number of parameters which makes the model more efficient and prevents overfitting. The most common type of pooling layer is a max-pooling layer, which takes the max from each down sampled block i.e. (2,2) in our model. We have chosen **adam optimizer** as the optimizer and **binary crossentropy** as the loss function. The figure below provides the summary of the model.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               802944
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 129
=================================================================
Total params: 813,217
Trainable params: 813,217
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: Binary Classifier Model Summary

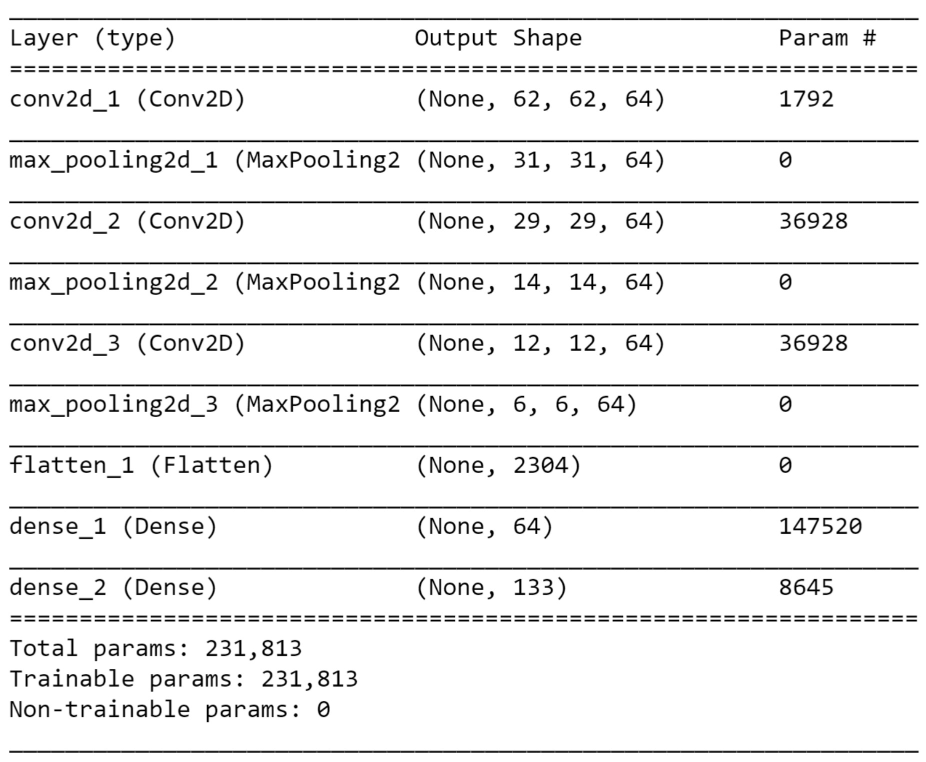
We have saved the trained model and it’s weights using the **ModelCheckpoint** from keras library and also we have implemented two utility methods **save_bclass()** and **load_bclass()** which are used to save and load the trained model, this save model can be reused to test or use the model.

We have the **isDog()** method which accepts the relative path of a image and predicts if it is dog or not dog. The method uses **imread** from skimage.io library to convert the image to a matrix and resize of the image to (64,64) to match the training size specified earlier is done by using **resize** method from skimage.transform library.

3.2 MultiClass Classifier

This is the main module of this project, which classifies the dog to it’s breed set of 133 classes. After detecting the image as image of a dog, the next step is to classify a dog according to it’s breed. So, for this we have built a multiclass classifier model. We have used the dataset from [6], it has around 6500+ images belonging to 133 breeds of dogs as training data and 800+ images belonging to 133 breeds of dogs ass test data. The algorithms performance is highly dependent on the dataset, so we have applied data augmentation by randomly shifting or zooming or horizontally flipping images in our dataset using keras **ImageDataGenerator**. This increases the size of the training set, which will increase accuracy by decreasing overfitting.

This model is like that of binary classifier with variations in the number of convolution layers and hidden layers. To build the CNN model for this classifier we have used Sequential class from Keras models, we have three layers of convolution each paired with a Pooling layer and then the extracted features are flattened (horizontally stacked) and fed into the dense fully connected layer. Here we have 1 fully connected layer with 64 units and the last layer acts as output layer with 133 unit. The convolution layer runs a sliding window through the image and at each step convolves the subimage with several filters i.e. each convolution layer extracts 64 features by using a stride (sliding window) of (3, 3), **relu** activation function is used to remove the negative values from these extracted features. To the first convolution layer the input image is formatted to the size (64,64) and since all the images are colored i.e. consists of RGB values we the complete size of the matrix becomes (64,64,3). The pooling layer reduces the size of the representation and the number of parameters which makes the model more efficient and prevents overfitting. The most common type of pooling layer is a max-pooling layer, which takes the max from each down sampled block i.e. (2,2) in our model. We have chosen **adam optimizer** as the optimizer and **categorical crossentropy** as the loss function. The figures below provide the architecture of the model and summary of the model.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 62, 62, 64)        1792
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 29, 29, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                147520
_________________________________________________________________
dense_2 (Dense)              (None, 133)               8645
=================================================================
Total params: 231,813
Trainable params: 231,813
Non-trainable params: 0
_________________________________________________________________
```

Figure 5: Multi-Class Classifier Model Summary

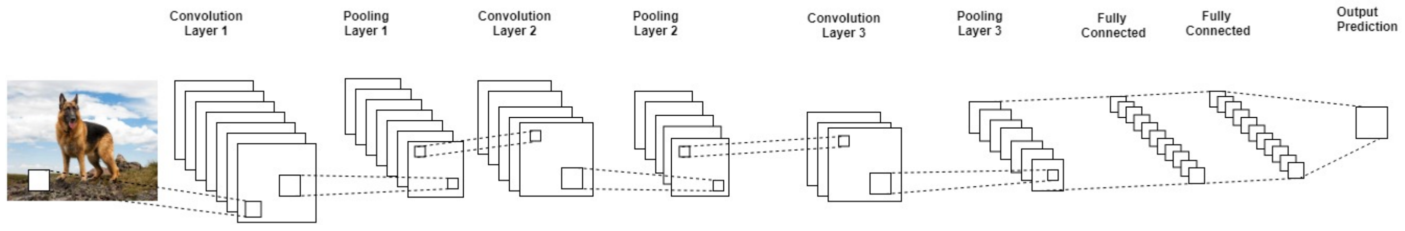


Figure 6: Multiclass CNN Architecture

We have saved the trained model and it’s weights using the **ModelCheckpoint** from keras library and also we have implemented two utility methods **save_mclass()** and **load_mclass()** which are used to save and load the trained model, this save model can be reused to test or use the model.

We have the **identifyBreed ()** method which accepts the classifier and relative path of a image and predicts the breed of the dog. The method uses **imread** from skimage.io library to convert the image to a matrix and resize of the image to (64,64) to match the training size specified earlier is done by using **resize** method from skimage.transform library.

3.3 Integration

To integrate both the models we have a utility method **whichDog()** which uses the **load_bclass()** and **load_mclass()** methods mentioned earlier to load the trained models and uses **isDog()** method to determine if the image is of a dog, if yes it continues execution and calls **identifyBreed()** method else it outputs the message that image is not of dog and stops execution. The **identifyBreed()** method predicts the breed of the dog and stops execution.

4 Experiment

The binary classifier was trained for 12 epochs with 22700 images for training set and 1461 images for test set. The training time for this model was around 3hrs for 2 epochs. The model gave an average accuracy of **99%** for both training data and test data. The loss is around **0.009** for train data and **0.02** to test data, these numbers clearly tell that the models performance is good.

The model was then tested with few random images of humans and other animals, it correctly classified them as not dog but for few instances images of other animals were confused as dog.

The multiclass classifier was trained for 50 epochs with 6680 images for training set and 836 images for test set. The training time for this model was around 10hrs for 5 epochs. For this the training accuracy is around **90 %** and loss has decreased from **3.5 to 0.4**. But the validation accuracy is still not stable and fluctuates around **14%** which is in line with the survey that we had done. And the validation loss keeps increasing as we train, maybe it needs more training where it starts to decrease after hitting a certain threshold. Or may be the model is over fitted and we must implement drop outs to reduce the loss, but taking the time left in the project into consideration we were not be able to do it, as we will lose the training that we did so far i.e. around 100 hrs.

The below figure summarizes the performance of the model.

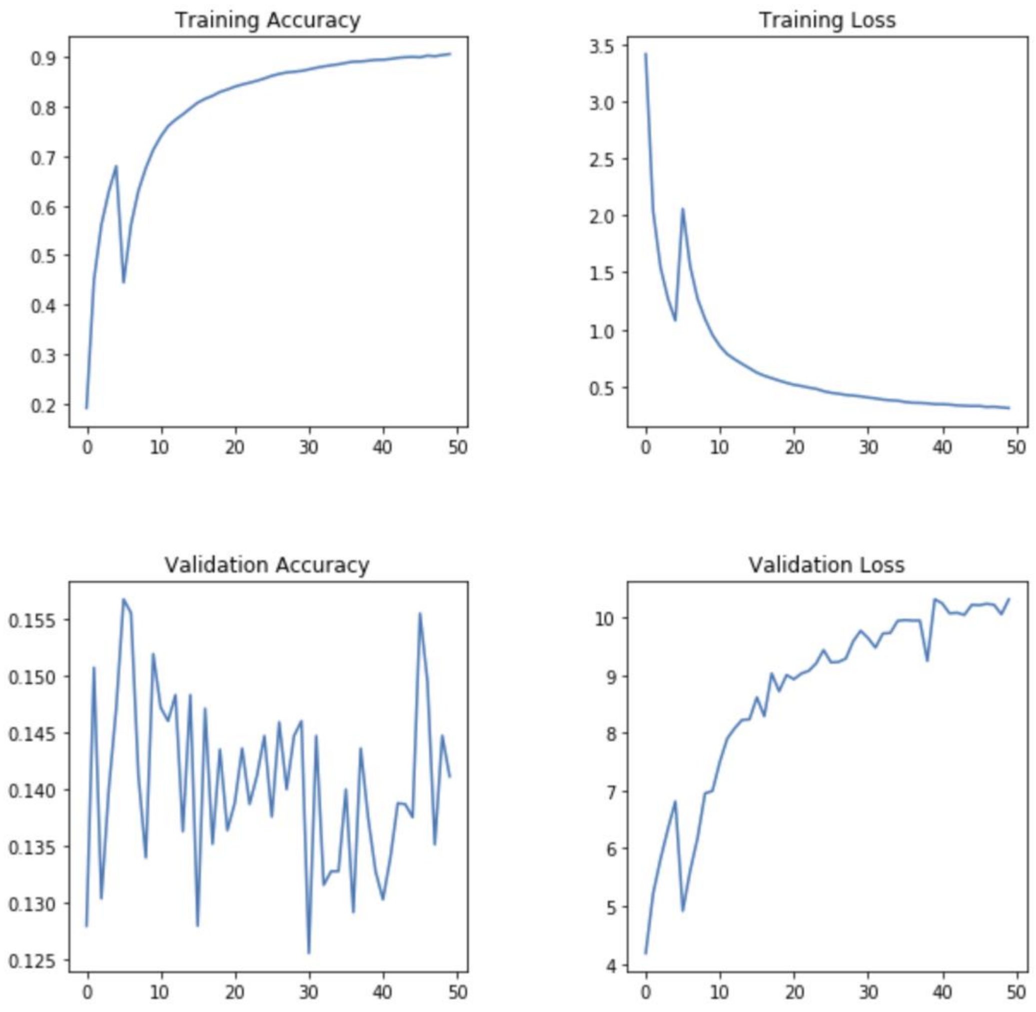


Figure 7: Multi-class Classifier performance history

Because of low accuracy of this model, the breeds identified by the model where not accurate but was classifying it to a breed which looks almost the same. For example, **Affenpinscher** was identified as **Belgian sheepdog** and **Belgian sheepdog** was identified as **Poodle**.

Affenpinscher	Belgian sheepdog	Poodle

5 Conclusion

This project proposed a method to classify dog breeds from various dog images by chaining a binary classifier and multiclass classifier. Because of the fine-grained nature of the images the accuracy achieved was not impressive but considering the training done and the models layers and their properties we achieved a not so bad accuracy of around **14%** in correctly classifying a new data. The proposed approach i.e. classifier chains is worth evaluating further as it reduced training times to greater extent and allowed the models to be separable and to be reused further. There were also restrictions on the number of features extracted and the convolution and fully connected layers used because they increase the training time significantly (1 extra dense layer almost contributed to a factor of 2 on the training time). Though building CNN from scratch is effective when it consists of ample number of layers and features extracted and trained sufficiently, it still is computationally too expensive.

Overall it was a great experience working on image recognition domain. We learnt a lot about CNN and how they work. We learnt what feature extraction is, how the filters are applied on the images. We also learnt what does convolution and pooling means. We also understood the importance, advantages and disadvantages of training the model. And of course, we did learn a lot about dog breeds and how to handle fine-grain classification.

6 Response to the feedback

6.1 Proposal feedback

Dog breed classification is a known problem. What can make your project different others? Think hard about this when you develop your project and reflect it on your mid/final report. Proposal lacks in survey. Just listing ideas are not proper proposal. Based on research, you should have delved into the questions like what you can do, what exactly you will do, how subtasks are assigned and so on..

The response to this is in the summary of the approach.

6.2 Mid report feedback

I can see your deliberation to make this project more interesting. The modified idea sounds reasonable but a bit worried about the timeline. Don't forget to store the learned model to reuse it for test. If not, you will lose a lot of time that you spent for training.

We have saved the models as and when we kept training it, but we did not store the history of the models for the metric evaluation. We did a work around for that and later rectified the mistake.

7 Future Work

In future work, we hope to build a more complex CNN architecture and train them more to achieve better performance. We also would like to chain transfer learning models and explore those models on fine-grain classifications. Once sufficient accuracy is obtained we would like to test those models on other datasets which can contribute more towards the betterment of the society. We would also like to host this trained model and expose an API for public use.

8 References

[1] Alex Krizhevsky. Ilya Sutskever and Geoffrey E. Hinton - ImageNet Classification with Deep Convolutional Neural Networks.

[2] V.P.Gladis Pushpa Rathi and Dr.S.Palani - Brain Tumor MRI Image Classification With Feature Selection And Extraction Using Linear Discriminant Analysis.

[3] Pratik Devikar - Transfer Learning for Image Classification of various dog breeds.

[4] Whitney LaRow , Brian Mittl, Vijay Singh - Dog Breed Identification.

[5] Jesse Read, Bernhard Pfahringer, Geoff Holmes, Eibe Frank - Classifier Chains for Multi-Label Classification

[6] Dataset: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip

[7] CNN Fundamentals: https://www.jeremyjordan.me/convolutional-neural-networks/

[8] Keras Documentation: https://keras.io/layers/convolutional/